AI Engineering Internship Technical Assessment

ITNB AG · Suurstoffi 37 · 6343 Rotkreuz · Switzerland · +41 41 790 11 90 · backoffice@itnb.ch · itnb.ch

Table of Contents

Overview

This technical assessment is designed to evaluate your practical skills in API integration, basic text-based retrieval (RAG), and your critical thinking in relation to secure access and system design at scale. You will first complete a hands-on Python exercise that integrates with the GroundX API, involving web scraping, data ingestion, and a headless Q&A "widget" (no UI). This is followed by two written design questions assessing your approach to access control and large-scale RAG architecture.

Stage 1: GroundX Integration, Web Scraping & Headless Chat

Objective: Demonstrate your ability to integrate GroundX with an automated data pipeline by scraping, ingesting, and querying real website content; simulating a "website widget" without a UI.

P.S. You are welcome to use AI coding assistants or tools (e.g., Cursor, Claude Code, or similar) to support your work in Stage 1; this will not be viewed negatively. Therefore, if that's the case, please include proof of these interactions in your submission using whichever method makes the most sense: share a public chat link, export the chat transcript, or include screenshots if export is not available.

Instructions

1. **GroundX Setup**
    a. Follow the official GroundX documentation to create a test account and obtain your personal API key (you may use the trial environment as indicated in their setup guide).
    b. Use the provided LLM credentials for query and chat functions
    c. If you have any issues creating an account, reach out to us.
2. **Website Scraping**
    a. Use the open-source scraper crawl4ai
    b. Crawl and extract textual content exclusively from https://www.itnb.ch/en
    c. Clean, structure, and store the website data locally for ingestion
3. **Content Ingestion**
    a. Programmatically ingest the scraped website content into GroundX using the Ingest API.
    b. The script should log and report success or failure for each ingested item.
4. **Chat Session Q&A**
    a. Write a Python script (or extend the above) that offers a simple command-line ("chat-like") prompt where the user can type a question.
    b. When a user asks a question, your script should:

i. Send the query to the GroundX Search API.
ii. Retrieve and display the top-matching chunk or answer from the scraped ITNB content.
iii. Include the source URL or title for context.

Deliverables for Stage 1:

- Submit a GitHub repository containing:
  - Python scripts for scraping, ingestion, and chat (.py).
  - A short README with setup and usage instructions.
  - Screenshots showing:
    - Successful ingestion of website content.
    - At least two Q&A interactions in the terminal proving functionality
- If AI Code Assistants were used, include proof of interactions along with your submission.

Outcome:
By the end of this stage, you will have built a minimal end-to-end RAG pipeline; scraping real web data, indexing it into GroundX, and querying it through a headless text-based interface.

Stage 2: Written Design Questions

Objective: Demonstrate your system design thinking for real-world, secure, and scalable enterprise RAG deployments. You are free to perform online research for the questions in this section. Please provide concise written answers (200–500 words each).

Question 1: Role-Based Access Control (RBAC) Implementation

Imagine an enterprise scenario where different users are assigned differing levels of document access rights in their file management system (e.g. MS SharePoint). For example, some users can see only their department's documents, while others (e.g., managers) can access a wider set.

Based on the GroundX documentation and available API capabilities, how would you design a document-level access control (authentication/authorization) mechanism?

Please address:

- How you would associate documents and users (e.g., using metadata, buckets, or tags)
- How you would enforce access restrictions at query time
- What limitations or security considerations you foresee

Question 2: Scaling RAG for Large and Dynamic Knowledge Bases

At ITNB, we offer a product named Sovereign Orchestrator. It is enterprise-grade, sovereign AI concierge that unifies search and decision support across sensitive, fragmented data. It delivers trusted, multilingual answers via natural language, integrates with enterprise tools (e.g., MS365), and orchestrates multi-step workflows while enforcing CH data residency, governance, and RBAC.

Corporate clients often need to manage thousands of documents, with frequent updates, additions, and deletions. Efficient, up-to-date retrieval is critical for a positive user experience.

How would you architect a scalable RAG solution in the context of an AI Concierge, focusing especially on document management and user experience?

Please discuss:

- Mechanisms you would use to handle large-scale, frequently changing document sets
- Whether you would empower users to manage documents themselves (and if so, how and why)
- Any algorithms, automation, or agent workflows you would design to keep retrieval both efficient and up-to-date

LLM Credentials

- OPENAI_MODEL_NAME=openai/inference-llama4-maverick
- OPENAI_API_BASE=https://maas.ai-2.kvant.cloud
- OPENAI_API_KEY=sk-7J_7Aag0iQdX_yLaYn90Zg